韩山师范学院实验报告

姓名：林婷婷　　专业：信息管理与信息系统　班级：20181171　　学号：2018117126

科目：　Android 应用开发　　实验日期：2021.6.11

实验题目：详解广播机制

【实验目的】

全局大喇叭，详解广播机制

【实验环境及方式】

Android Studio

【实验内容及实验结果】（备注：结果可以采用界面截图）

1. 广播机制简介

为了便于进行系统级别的消息通知，Android 引入了一套广播消息机制。

每个应用程序都可以对自己感兴趣的广播进行注册，这样该程序就只会收到自己所关心的广播内容，这些广播可能是来自于系统的，也可能是来自于其他应用程序的。Android 提供了一套完整的 API，允许应用程序自由地发送和接收广播。

Android 中的广播主要可以分为两种类型：标准广播和有序广播。

标准广播（normal broadcasts）是一种完全异步执行的广播，在广播发出之后，所有的 BroadcastReceiver 几乎都会在同一时刻接收到收到这条广播消息，因此它们之间没有任何先后顺序可言。这种广播的效率会比较高，但同时也意味着它是无法被截断的。

有序广播（ordered broadcasts）是一种同步执行的广播，在广播发出之后，同一时刻只会有一个 BroadcastReceiver 能够收到这条广播消息，当这个 BroadcastReceiver 中的逻辑执行完毕后，广播才会继续传递。所以此时的 BroadcastReceiver 是有先后顺序的，优先级高的 BroadcastReceiver 就可以先收到广播消息，并且前面的 BroadcastReceiver 还可以截断正在传递的广播，这样后面的 BroadcastReceiver 就无法收到广播消息了。

2. 接收系统广播

开发者可以根据自己感兴趣的广播，自由地注册 BroadcastReceiver，这样当有相应的广播发出时，相应的 BroadcastReceiver 就能够收到该广播，并可以在内部进行逻辑处理。

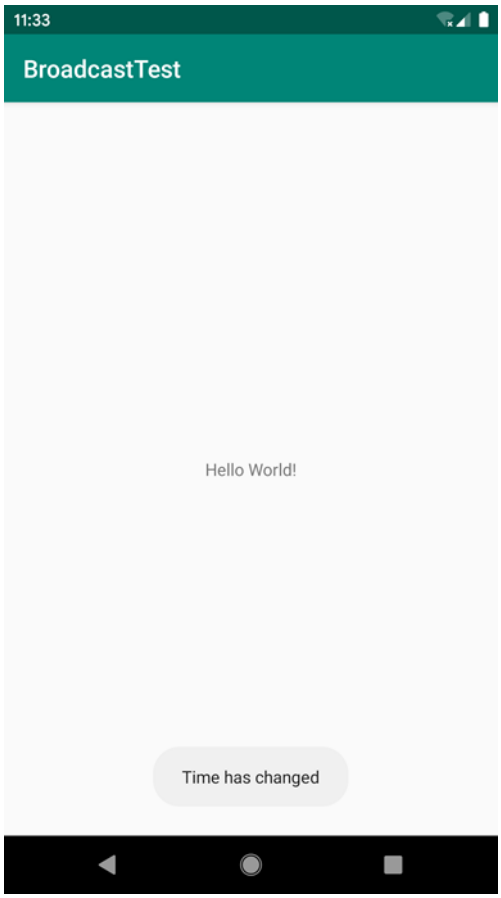
注册 BroadcastReceiver 的方式一般有两种：在代码中注册和在 AndroidManifest.xml 中注册。其中前者也被称为动态注册，后者也被称为静态注册。

① 动态注册监听时间变化

动态注册的代码示例如下：

```
class MainActivity : AppCompatActivity() {
lateinit var timeChangeReceiver: TimeChangeReceiver
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)
```

```
val intentFilter = IntentFilter()
intentFilter.addAction("android.intent.action.TIME_TICK")
timeChangeReceiver = TimeChangeReceiver()
registerReceiver(timeChangeReceiver, intentFilter)
}
override fun onDestroy() {
super.onDestroy()
unregisterReceiver(timeChangeReceiver)
}
inner class TimeChangeReceiver : BroadcastReceiver() {
override fun onReceive(context: Context, intent: Intent) {
Toast.makeText(context, "Time has changed", Toast.LENGTH_SHORT).show()
}
}
}
```



② 静态注册实现开机启动

静态注册的代码示例如下：

```
class BootCompleteReceiver : BroadcastReceiver() {
```

```
        override fun onReceive(context: Context, intent: Intent) {
            Toast.makeText(context, "Boot Complete", Toast.LENGTH_LONG).show()
        }
    }
```

另外，静态的 BroadcastReceiver 一定要在 AndroidManifest.xml 文件中注册才可以使用：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.broadcasttest">

<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <receiver
        android:name=".BootCompleteReceiver"
        android:enabled="true"
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.BOOT_COMPLETED" />
        </intent-filter>
    </receiver>
</application>
</manifest>
```

3. 发送自定义广播

① 发送标准广播

构建一个 Intent 对象，并把要发送的广播的值传入。然后调用 Intent 的 setPackage()方法，并传入当前应用程序的包名。最后调用 sendBroadcast()方法将广播发送出去，这样所有监听 com.example.broadcasttest.MY_BROADCAST 这条广播的 BroadcastReceiver 就会收到消息了。此时发出去的广播就是一条标准广播。

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        button.setOnClickListener {
            val intent = Intent("com.example.broadcasttest.MY_BROADCAST")
            intent.setPackage(packageName)
```

```
                sendBroadcast(intent)
            }
        }
    }
```

② 发送有序广播

和标准广播不同，有序广播是一种同步执行的广播，并且是可以被截断的。

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        button.setOnClickListener {
            val intent = Intent("com.example.broadcasttest.MY_BROADCAST")
            intent.setPackage(packageName)
            sendOrderedBroadcast(intent, null)
        }
    }
}
```

发送有序广播只需要改动一行代码，即将 sendBroadcast()方法改成 sendOrderedBroadcast()方法。

③ 接收自定义广播

在发送广播之前，我们还是需要先定义一个 BroadcastReceiver 来准备接收此广播，不然发出去也是白发。因此新建一个 MyBroadcastReceiver，并在 onReceive()方法中加入如下代码：

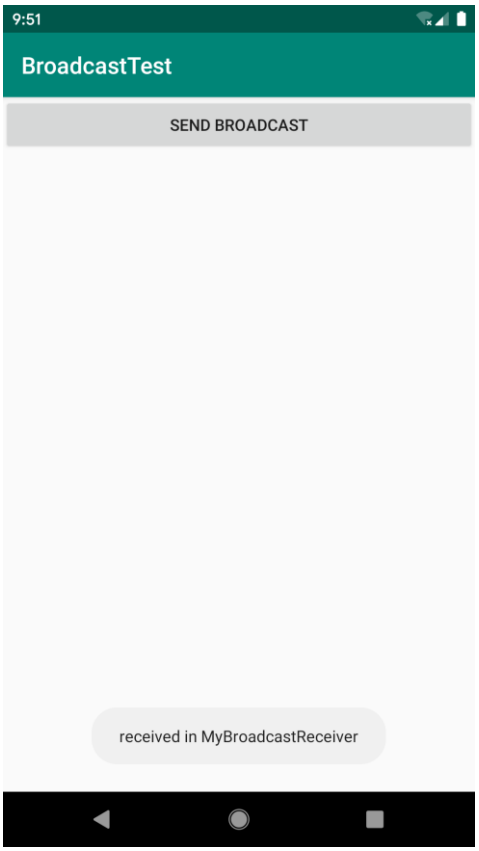
```
class MyBroadcastReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        Toast.makeText(context, "received in MyBroadcastReceiver",
    Toast.LENGTH_SHORT).show()
    }
}
```

然后在 AndroidManifest.xml 中对这个 BroadcastReceiver 进行注册：

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.broadcasttest">
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
```

```
    …
    <receiver
        android:name=".MyBroadcastReceiver"
        android:enabled="true"
        android:exported="true">
        <intent-filter>
            <action android:name="com.example.broadcasttest.MY_BROADCAST"/>
        </intent-filter>
    </receiver>
</application>
</manifest>
```

点击按钮后收到广播：



【教师评语和成绩】

成绩： **指导教师：** **日期：**